UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/687,092	10/12/2000	Andrew E. Blau	CA9-1998-0006	9550

7590 08/26/2009

David A. Mims, Jr.
International Business Machines Corporation
Intellectual Property Law Department
Internal Zip 4054, 11400 Burnet Road
Austin, TX 78758

EXAMINER	
CHUONG, TRUC T	

ART UNIT	PAPER NUMBER
2179	

MAIL DATE	DELIVERY MODE
08/26/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte ANDREW E. BLAU
and EDUARDUS A.T. MERKS

Appeal 2008-002919
Application 09/687,092[1]
Technology Center 2100

Decided: August 26, 2009

Before LEE E. BARRETT, JEAN R. HOMERE, and JOHN A. JEFFERY, *Administrative Patent Judges*.

BARRETT, *Administrative Patent Judge*.

DECISION ON APPEAL

---

[1] Filed October 12, 2000, titled "System and Method for Managing Messages and Annotations Presented in a User Interface," which claims the benefit of Canada Application 2,293,068, filed December 22, 1999.

This is a decision on appeal under 35 U.S.C. § 134(a) from the final rejection of claims 1-5, 7-22, and 24-30. Claims 6 and 23 are objected to as depending from a rejected base claim. We have jurisdiction pursuant to 35 U.S.C. § 6(b).

We affirm.

STATEMENT OF THE CASE

The invention

The invention relates to the presentation, in a graphic user interface, of compiler error messages together with user-supplied annotations. Spec. 1. The invention is well described in the Appeal Brief at pages 2-4.

During application code development, once code is created, it must be compiled to determine if the code contains errors. If there are errors, the compiler will generate error messages to help the code developer diagnose the coding errors. Appellants' invention adds a new step to the process by providing user annotations to the compiler error messages. Programmers working with the same compilers over extended periods of time frequently come to recognize certain error messages, and know what the likely cause and solution to the problems which caused the error messages. This information remains in the developer's mind unavailable for assisting other developers who encounter the same error messages. In addition, programmers recognize a smaller subset of error messages, and have their interpretations that are specific to the quirks of a particular project.

Appellants' invention provides a means to allow other code developers to take advantage of this knowledge by allowing user annotations to compiler error messages. A second, initially empty, error message file is created and maintained for each error message generated by the compiler.

Spec. 13, l. 12. When a code developer adds a note to the compiler error message, the user's note is stored in the second error message file. As shown in Figure 2, compiler error messages 53 are textual material stored in a message catalog 50. The compiler error messages are identified and accessed by way of message keys 52 for display to a code developer in a window on a user's workstation. Message catalog name 51 is combined with message keys 52 to generate unique annotation dialog keys 54 which are used to access the annotations 56 for each compiler error message stored in the separate file. Appellants' invention allows this separate annotation file to be transmitted or copied to another code developer's workstation and when the compiler generates the error message, the appropriate annotation will be displayed thus allowing multiple code developers to share experiences.

Representative claim

Claim 1 is reproduced below:

1. A method for managing compiler error messages, comprising the steps of:

displaying a compiler error message having a separate empty error file to a user;

accepting from said user an annotation to said compiler error message;

associating said annotation with said compiler error message using a unique key and storing said annotation in said separate empty error file; and

thereafter selectively displaying said annotation with said compiler error message.

The references

Mueller	5,673,390	Sep. 30, 1997
Hughes	US 6,275,223 B1	Aug. 14, 2001 (filed July 8, 1998)

The rejections[2]

Claims 1-5, 7-22, and 24-30 stand rejected under 35 U.S.C. § 103(a) as unpatentable over Mueller and Hughes.

Grouping of claims

Appellants group the claims as follows:

1. Claims 1-5, 7-12, 16-22, and 24-27;
2. Claims 13-15, 29, and 30; and
3. Claim 28.

FACTS

Mueller

Mueller discloses a technique for interactively displaying error messages, such as parser or compiler messages, associated with a user's source code. Col. 1, ll. 61-64.

Mueller discloses in the background that compilers typically provide programmers with a compiler listing which lists the source code along with the errors. Col. 1, ll. 29-31. Mueller teaches that these errors maybe listed at the end of the source listing, interspersed through the source listing, or in some instances the errors are displayed in the same window in which the source file is being edited. Col. 1, ll. 31-36.

[2] The rejection of claim 13 under 35 U.S.C. § 101 and the rejection of claims 1-12 under 35 U.S.C. § 112 ¶ 2 have been withdrawn. Ans. 2.

Mueller specifically addresses special problems where compiler errors code emanate from multiple sources, such as a local parser or compiler, as well as a remote compiler which runs on a second computer, such as a mainframe host computer connected through a communication link. Col. 1, ll. 52-56.

Mueller describes that "[t]he language processors create an Events File which contains the information identifying the file in which the error occurs, the error types and the locations of the error in the file when possible." Col. 2, l. 66 to col. 3, l. 2.

"The Events File is processed by the Events File Processor which generates a set or [sic, of] program-to-program messages which are sent to the Error List Processor which builds the Error List." Col. 3, ll. 5-8.

The Error List contains a list of all errors and is displayed as a list of files containing errors and the error messages in those files in a read-only window as shown in Figure 3. Col. 5, ll. 33-39. This allows the user to see all errors in one location (the Error List) while still being able to easily locate the file and text containing the error. Col. 3, ll. 37-40.

Each file in the Error List window has a pointer to a File Information Record (FIR) and each message in the Error List window has a pointer to an Error Information Record (EIR). Col. 8, ll. 20-23.

The Error List is modified after parsing, after compiling, after a Get events file action, after deleting text corresponding to an error, and after changing text corresponding to an error. Col. 8, l. 25 to col. 9, l.7.

Hughes

Hughes discloses a code inspection tool for allowing developers to compile annotation data and forward the annotation data from a graphical user interface to a centralized data store. Col. 3, ll. 57-60.

Hughes discloses that during a code inspection session, individual lines of code are inspected by all or some of the developers on a line-by-line basis, with each developer making notes of any proposed changes to the portions of code for which they have responsibility and the changes being recorded by a moderator. Col. 2, ll. 24-31. This process is said to have problems, such as having to bring code to the meetings and preparation of minutes of the meeting. Col. 2, l. 43 to col. 3, l. 26.

Hughes discloses an invention where during code inspection sessions, all developers view a display, which sets out original source code, side by side with new source code on a line by line basis, matching the line numbers of the original source with those of the new source code. Col. 3, ll. 62-67.

Figure 15 of Hughes shows a display window in accordance with the invention. Original source code is displayed in window 1501 and new (amended) source code is displayed in window 1502, such that lines of original source code are directly comparable to corresponding lines of new source code, shown in line number windows 1503, 1504. Col. 12, ll. 23-61.

During code inspection, modifications to the source code can be entered and annotations to the code can be made. Col. 9, ll. 59-61. Annotations to the source code lines are indicated by icons next to the appropriate lines. Col. 4, ll. 3-5.

Developers may prepare annotations to original source code by entering annotation data into annotation data files in a single file inspection

window as shown in Figure 17, using an edit annotation window as shown in Figure 18. Col. 14, ll. 1-57. Hughes describes an "edit annotation box through which the developer can enter annotation data to a separate file." Col. 14, ll. 16-19. Hughes describes that "received annotation files are logically matched up with the line numbers of the original source code, and annotation icons are generated in annotation window 1508 adjacent [to] the line to which the respective annotation line belongs." Col. 14, ll. 27-30.

Hughes describes that annotations may be viewed by selecting an annotate view item from an annotate menu icon which causes the computer to generate a view annotations display as shown in Figure 19. Col. 15, ll. 8-11. The annotation data may be edited using the view annotations window display. Col. 15, ll. 12-13.

Hughes describes that annotations may be of three types: a comment on a corresponding line of code; an indication that a change or error in the code does not affect operation of the functionality of the code; and an indication that an error in a line of code does significantly alter the operation of a portion of the source code as a whole. Col. 15, ll. 8-29.

"Modifications of the best mode described herein include interfacing the code inspection tool with such known automatic error detection tools [as the ProLint database]." Col. 16, ll. 26-28.

PRINCIPLES OF LAW

"[T]he test [for obviousness] is what the combined teachings of the references would have suggested to those of ordinary skill in the art." *In re Keller*, 642 F.2d 413, 425 (CCPA 1981) (citations omitted). A rejection under 35 U.S.C. § 103(a) is based on the following factual

determinations: (1) the scope and content of the prior art; (2) the level of ordinary skill in the art; (3) the differences between the claimed invention and the prior art; and (4) any objective indicia of non-obviousness. *DyStar Textilfarben GmbH & Co. Deutschland KG v. C.H. Patrick Co.*, 464 F.3d 1356, 1360 (Fed. Cir. 2006) (citing *Graham v. John Deere Co.*, 383 U.S. 1, 17 (1966)). Whether there is motivation to combine or modify the references is a question of fact drawing on the factors of *Graham*. *See McGinley v. Franklin Sports, Inc.*, 262 F.3d 1339, 1351-52 (Fed. Cir. 2001). "[H]owever, the analysis need not seek out precise teachings directed to the specific subject matter of the challenged claim, for a court can take account of the inferences and creative steps that a person of ordinary skill in the art would employ." *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 418 (2007).

ANALYSIS

Claims 1-5, 7-12, 16-22, and 24-27

The Examiner finds that the Error List displayed in Figure 3 is a separate file, which can be empty if there is no error during compiling, and therefore Mueller teaches "displaying a compiler error message having a separate empty error file to a user." Final Rej. 4. The Examiner finds that "Mueller does not clearly point out that an edit panel is accepting from user as an annotation, associating the annotation with the compiler error message using a unique key, and storing the annotation in said separate empty file." Final Rej. 4. The Examiner finds that Hughes teaches an editing annotation window using a unique key (each annotation identifying a line number of code) and storing the annotations. *Id.* at 4-5. The Examiner concludes that it would have been obvious "that a person of ordinary skill in the art would

want to have the editable annotation data of Hughes in the Error List of Mueller to allow users to easily find and edit specific errors in a program code during compiling." *Id.* at 5.

Appellants argue that neither of the references teaches or suggests displaying a compiler error message having a separate empty error file into which an annotation is associated and stored. Br. 8.

We agree with Appellants that Mueller does not describe a separate empty error file. The limitation of "displaying a compiler error message having a separate empty error file to a user" calls for an "empty error file" which is "separate" from the "compiler error message." The Error List in Mueller contains the compiler error messages, which are displayed to a user and is not a *separate empty file* from the compiler error messages. Mueller also does not teach an annotation associated with a compiler error message. Mueller discloses compiler error messages without any annotations.

However, Hughes describes providing an annotation to a line or lines of source code. For example, in edit annotate window 1800 in Figure 18, a developer can add an annotation to lines 6 through 13 of source code, and annotations are indicated by icons next to the appropriate lines as shown in Figure 15. Hughes describes an "edit annotation box through which the developer can enter annotation data to *a separate file*." Col. 14, ll. 16-19 (emphasis added). Thus, Hughes discloses displaying lines of a source code file having a separate empty annotation file to a user. The issue is whether the references suggest providing annotations to *compiler error messages*.

Hughes describes that lines of the source code displayed in the code inspection tool can have errors identified by, for example, automatic error detection tools. Col. 15, l. 19 to col. 16, l. 31. Since developers can

annotate the code to say whether the error does or does not affect the code operation, col. 15, ll. 23-43, this reasonably suggests that the error message is associated with and displayed with the line of code. While Hughes does not describe that the errors are compiler errors, one of ordinary skill in the computer art would have known that compiler errors were a common (if not the most common) type of program error, as evidenced by Mueller, and would have been motivated to include compiler errors to the types of errors that can be annotated. Complier errors are associated with lines of program code. Appellants argue that Hughes is directed to source code inspection, which is far different from compiler error examination. Br. 7. However, Appellants do not address the teaching of annotating lines of source code having errors in Hughes. Of course, when the errors are eventually corrected in Hughes, the annotations will presumably disappear, whereas in Appellants' disclosed invention, the annotations remain associated with compiler error messages which can appear after compiling other programs. However, this permanence concept is not captured in claim 1.

The next issue is whether Hughes teaches or suggests "associating said annotation with said compiler error message using a unique key and storing said annotation in said separate empty error file." Hughes describes that the annotations are stored in a separate file. Col. 14, l. 17. Hughes describes that "[t]he received annotation files are logically matched up with the line numbers of the original source code, and annotation icons are generated in annotation window 1508 adjacent [to] the line to which the respective annotation line belongs. Col. 14, ll. 27-30. We agree with the Examiner's finding (at Final Rej. 4-5) that the line numbers are a unique key which associates the annotation to the source code. In addition, one of

ordinary skill in the art would have had sufficient skill to appreciate that there must be some key to associate an annotation to the associated line.

Appellants argue that providing an empty file for annotations would be impractical in Hughes because it would require an empty error file for each line of source code, which would require enormous storage requirements. It is argued that Applicants' invention, in contrast, is working only with the errors generated from compiler error diagnosis, which results in limiting annotations and storage to a limited set of errors. Br. 7-8.

This argument is not persuasive. Each annotation in Hughes is associated with one or more line numbers as shown in Figure 18. There is no indication that every line number has its own annotation file. Thus, there are only as many annotation files as there are lines to be annotated. In any case, impracticality does not preclude obviousness.

Appellants also argue that neither of the references discloses associating an annotation with a compiler error message using a unique key. It is argued that in Appellants' invention, message catalog name 51 (Figure 2) is combined with message keys 52 to generate unique annotation dialog keys 54, which are used to access the annotations 56 for each compiler error message stored in a separate file. Br. 8.

This argument is not persuasive because Appellants do not state why the line number(s) in Hughes cannot be considered a key.

For the reasons stated above, we conclude that Appellants have not shown error in the Examiner's rejection. The rejection of claims 1-5, 7-12, 16-22, and 24-27 is affirmed.

Claims 13-15, 29, and 30

Appellants argue that neither of the references teaches an "event driven control component for displaying using a unique key an associated annotation in said user display," as recited in claim 13. Br. 8. Appellants argue that "[i]n Hughes, the source code line numbers are used to associate an annotation with a particular line of source code." *Id.*

This argument does not explain why the line numbers in Hughes are not a unique key, as discussed in connection with claim 1. Accordingly, we conclude that Appellants have not shown error in the Examiner's rejection. The rejection of claims 13-15, 29, and 30 is affirmed.

Claim 28

Appellants argue that there is no teaching or suggestion in any of the references of the concept of "selectively displaying said annotation with said compiler error message" and "causing a computer to effect presenting an edit panel in said graphic user interface for user entry of new or modified annotations," as recited in claim 28.

As discussed in connection with claim 1, Hughes describes annotating lines of source code to indicate whether errors in the code do or do not significantly affect the operation of the code. Col. 15, l. 19 to col. 16, l. 31. We concluded that this reasonably suggests that an error message is associated with and displayed with a line of code and the annotation is associated with the error message for that line. We also concluded that it would have been obvious for the error message to be a compiler error message. Hughes discloses that an annotation icon is displayed for a line of source code having an annotation associated with it. Col. 13, ll. 52-60. The

annotation may be viewed on a display as shown in Figure 19 by activating a menu icon or by selecting the icon. Col. 15, ll. 8-18. It would have been apparent to one skilled in the art that this window can be located so that one can view the annotation and the error message for the line of code. The display in Figure 19 allows editing of the annotation.

For these reason, we conclude that Appellants have not shown error in the Examiner's rejection. The rejection of claim 28 is affirmed.

CONCLUSION

The rejection of claims 1-5, 7-22, and 24-30 under 35 U.S.C. § 103(a) is affirmed.

Requests for extensions of time are governed by 37 C.F.R. § 1.136(b). *See* 37 C.F.R. § 41.50(f).

AFFIRMED

erc

David A. Mims, Jr.
International Business Machines Corporation
Intellectual Property Law Department
Internal Zip 4054, 11400 Burnet Road
Austin, TX 78758